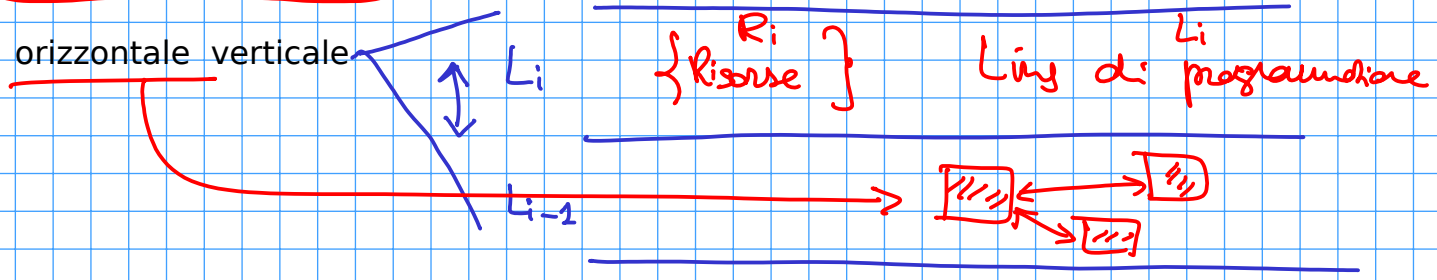
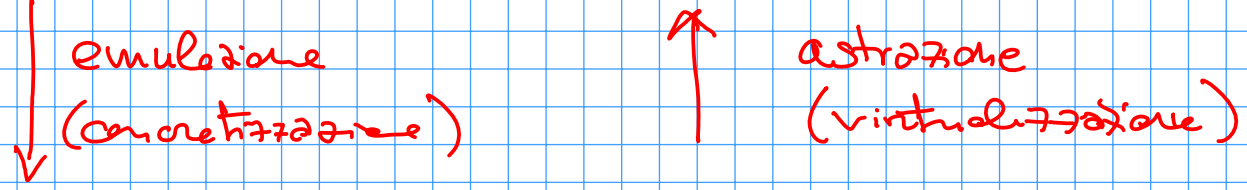
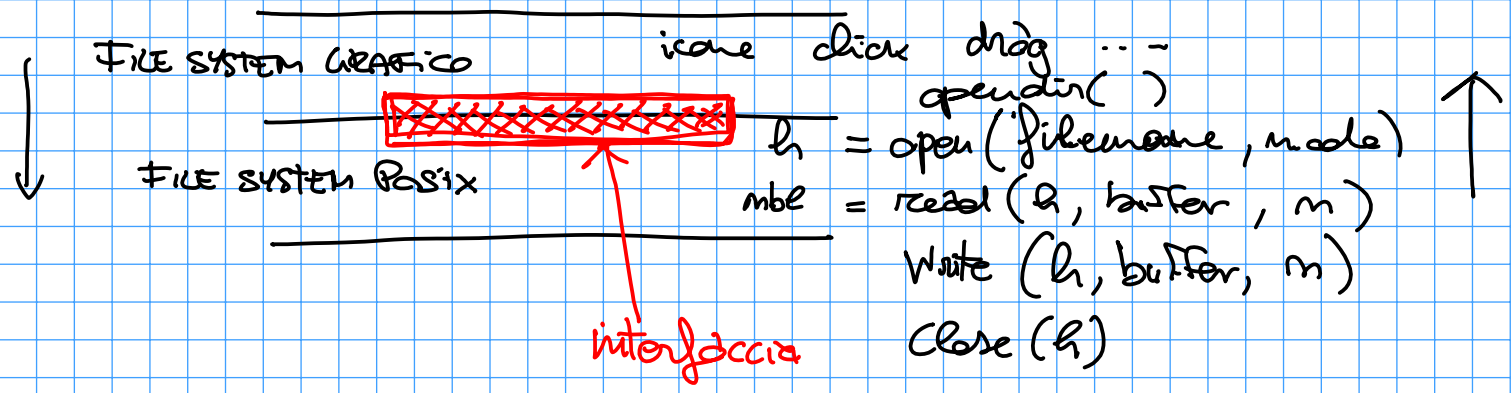
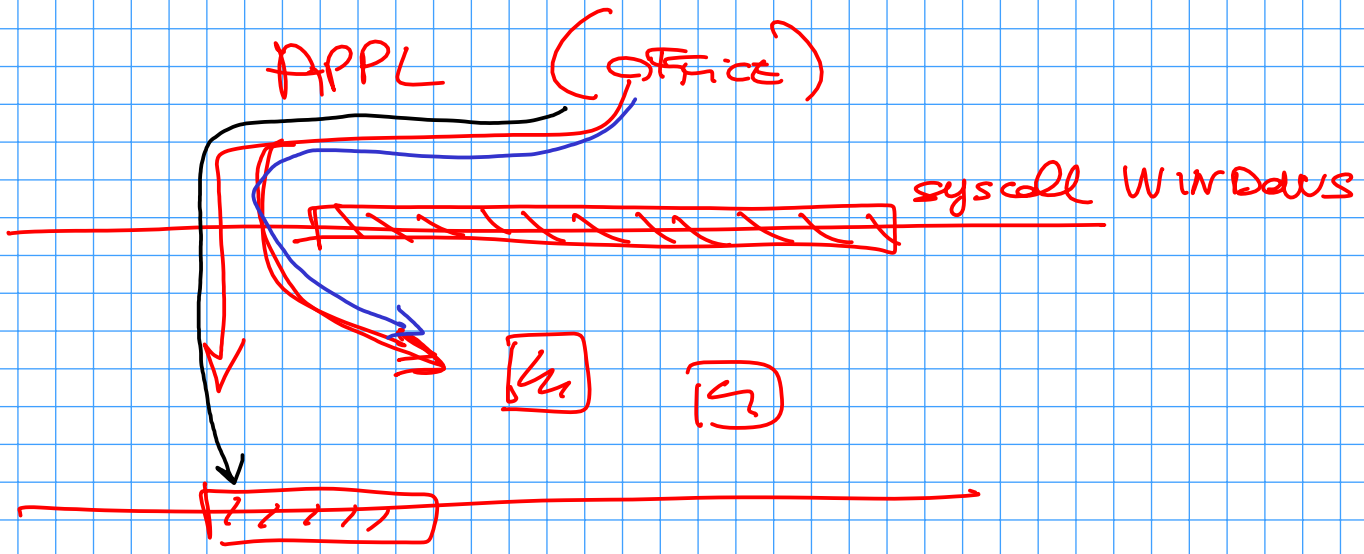
Strutturazione a livelli

orizzontale verticale









MV4 — APPLICAZIONI
R =
L = JAVA, C++/OCaml, C#, ASSEMBLER

MV3 — SISTEMA OPERATIVO
R = spazio di memoria, processore, I/O, programmi
L = Ling Prog (C) + chiamate di sistema

MV2 — MACCHINA ASSEMBLER
R = { locazioni di memoria, reg generali, I/O
L = linguaggio macchina

MV1 — MACCHINA FIRMWARE
R = { registri, memorie, operatori logici, ...
L = μ-programmazione

MV0 — MACCHINA HARDWARE
R = dispositivi elettronici L = ?

```
loop: LOAD  R_base , R_i , R_1
      LOAD  R_base2 , R_i , R_2
      ADD   R_1 , R_2 , R_1
      STORE R_base3 , R_i , R_1
      INC   R_i
      IF<   R_i , R_N , loop
      END
```

M[R[R_base] + R[R_i]] → R[1]

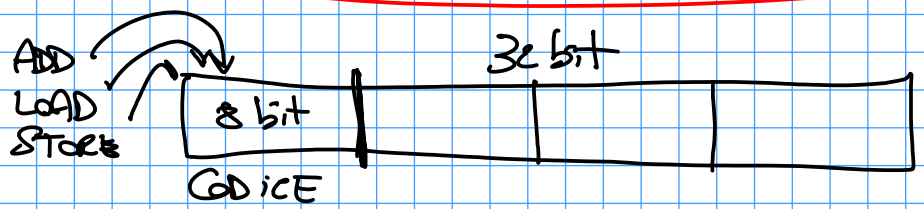
R[1] + R[2] → R[1]

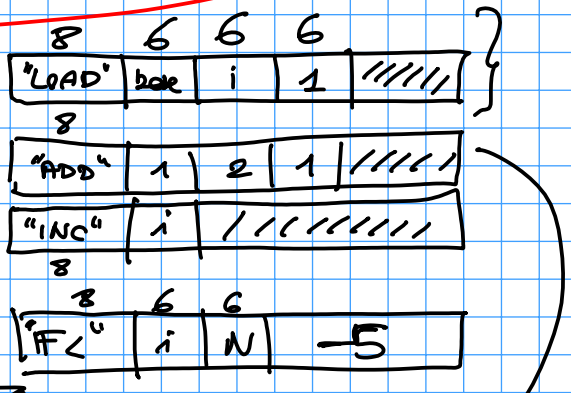
R_1 → M[R[base] + R[i]]

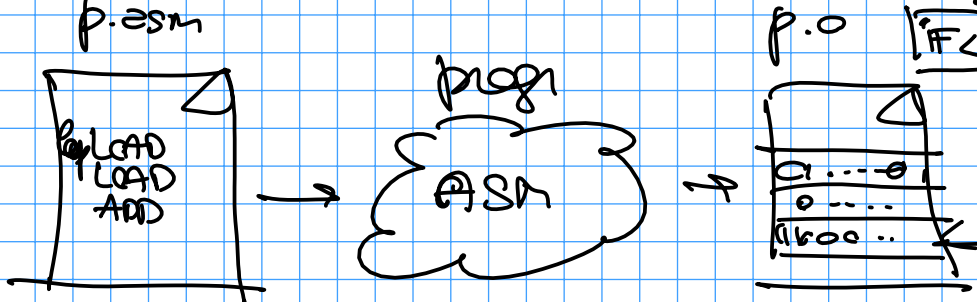
R_i + 1 → R_i

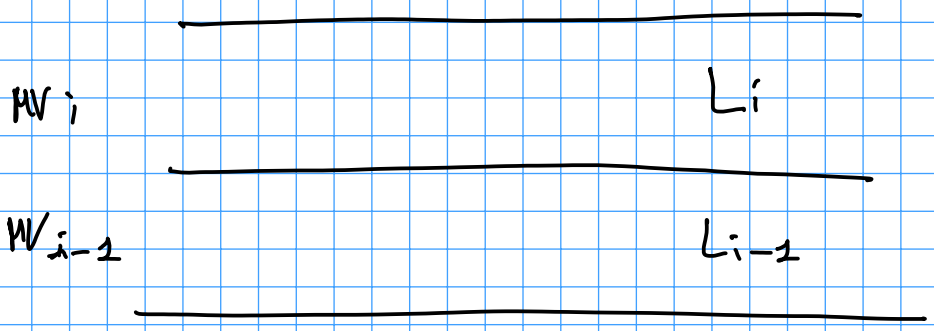
se R_i < R_N

else









compilazione

L_i → traduzione → L_{i-1}

(auce & forall)

Interpretazione

Legge L_i e lo interpreta (L_{i-1}) 1 istruzione alla volta

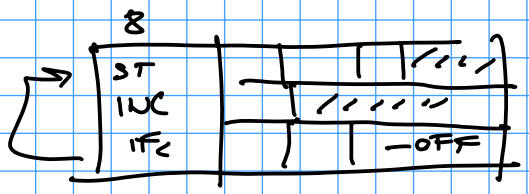
```
for (i=0 ; i<N ; i++) {
    x[i] = i;
}
```

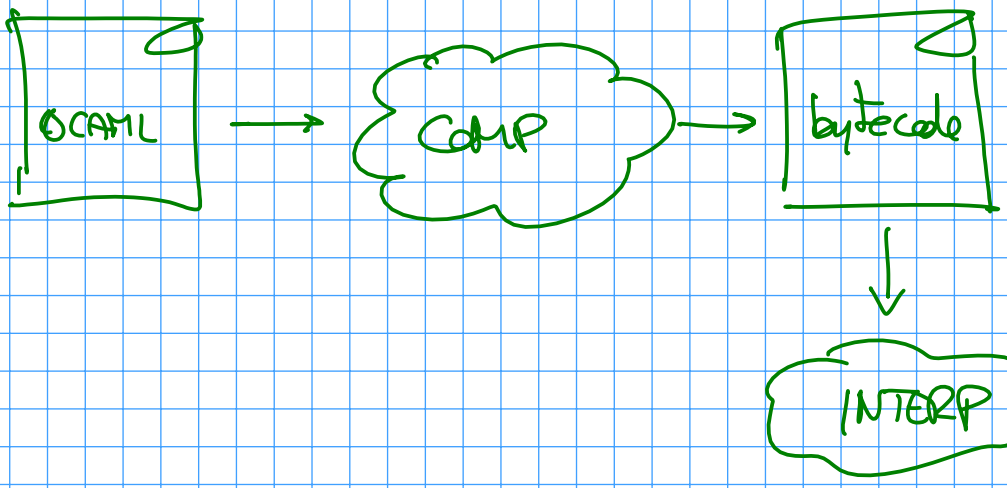
COMPILAZIONE

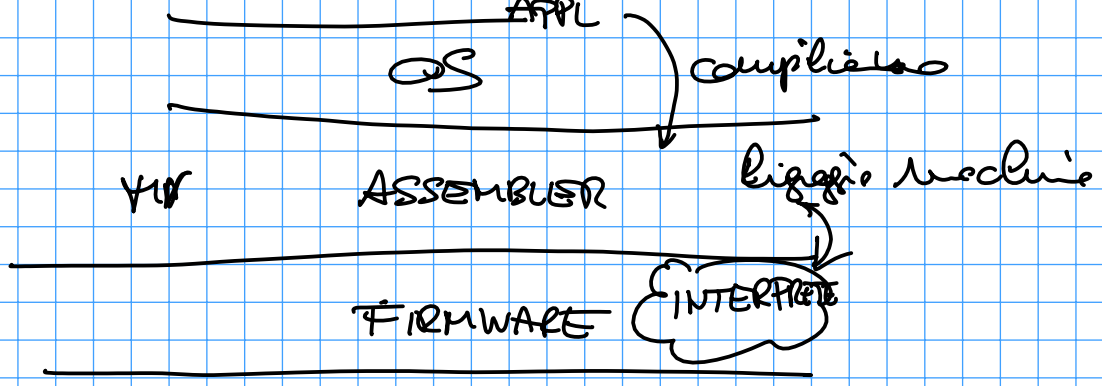
```
loop : STORE R_base, Ri, Ri
       INC  Ri
       IF< Ri, RN, loop
       ;
```

INTERPRETAZIONE

leggi riga interpreta esegui } LOAD






APPL
OS
compilatore
VHF
ASSEMBLER
linguaggio macchina
FIRMWARE
INTERPRETE